PRACTICAL: 4

AIM:

Encrypt a message using the standard Playfair cipher and attempt to decrypt it without the key by analyzing patterns in the ciphertext. Then, implement an extended Playfair cipher with a 10x10 matrix, incorporating uppercase/lowercase letters, digits, and symbols (e.g., @, #, $). Encrypt and decrypt a message with this extended cipher, demonstrating how the increased complexity improves security compared to the standard version.

THEORY:

Playfair Cipher

The Playfair cipher is a digraph substitution cipher that encrypts pairs of letters instead of single letters, making it more secure than simple monoalphabetic ciphers. It uses a 5x5 matrix of letters, constructed using a keyword, and follows specific rules for encryption and decryption.

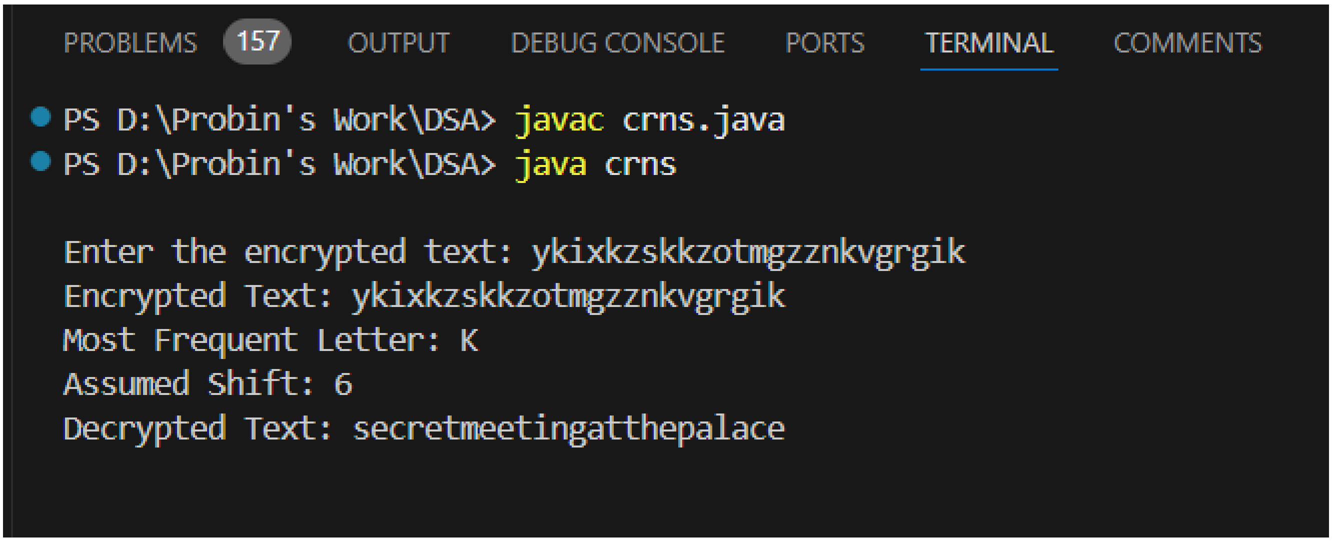
Encryption Rules:

1. If both letters in a pair appear in the same row, replace each with the letter to its right (wrapping around if needed).
2. If both letters are in the same column, replace each with the letter below it (wrapping around if needed).
3. If the letters form a rectangle, replace them with the letters on the same row but in the opposite corners.
4. If the plaintext contains repeated letters in a pair, an 'X' is inserted between them.
5. If the number of characters is odd, an extra 'X' is added at the end.

CODE:

```
import java.util.*;

public class crns {

    private static char[][] keyMatrix = new char[5][5];

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the key (without spaces): ");

        String key = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "").replace("J", "I");

        System.out.print("Enter the plaintext (without spaces): ");

        String plaintext = scanner.nextLine().toUpperCase().replaceAll("[^A-Z]", "").replace("J", "I");
        generateKeyMatrix(key);
        displayMatrix();
        String encryptedText = encryptText(plaintext);
        System.out.println("Encrypted Text: " + encryptedText);
        scanner.close();
    }
    private static void generateKeyMatrix(String key) {
        StringBuilder uniqueKey = new StringBuilder();
        boolean[] used = new boolean[26];
        for (char c : key.toCharArray()) {
            if (!used[c - 'A'] && c != 'J') {
                uniqueKey.append(c);
                used[c - 'A'] = true;
            }
        }
  for (char c = 'A'; c <= 'Z'; c++) {
            if (!used[c - 'A'] && c != 'J') {
                uniqueKey.append(c);
            }
        }
        int index = 0;
        for (int row = 0; row < 5; row++) {
            for (int col = 0; col < 5; col++) {
                keyMatrix[row][col] = uniqueKey.charAt(index++);
            }
        }
    }
    private static void displayMatrix() {
        System.out.println("Key Matrix:");
        for (int row = 0; row < 5; row++) {
            for (int col = 0; col < 5; col++) {
                System.out.print(keyMatrix[row][col] + " ");
            }
            System.out.println();
        }
    }
    private static String encryptText(String text) {
        StringBuilder preparedText = prepareText(text);
        StringBuilder encryptedText = new StringBuilder();
        for (int i = 0; i < preparedText.length(); i += 2) {
            char first = preparedText.charAt(i);
            char second = preparedText.charAt(i + 1);
            int[] pos1 = findPosition(first);
            int[] pos2 = findPosition(second);
            if (pos1[0] == pos2[0]) { // Same row
                encryptedText.append(keyMatrix[pos1[0]][(pos1[1] + 1) % 5]);
                encryptedText.append(keyMatrix[pos2[0]][(pos2[1] + 1) % 5]);
            } else if (pos1[1] == pos2[1]) { // Same column
                encryptedText.append(keyMatrix[(pos1[0] + 1) % 5][pos1[1]]);
                encryptedText.append(keyMatrix[(pos2[0] + 1) % 5][pos2[1]]);
            } else { // Rectangle swap
                encryptedText.append(keyMatrix[pos1[0]][pos2[1]]);
                encryptedText.append(keyMatrix[pos2[0]][pos1[1]]);
            }
        }
        return encryptedText.toString();
    }
    private static StringBuilder prepareText(String text) {
        StringBuilder preparedText = new StringBuilder(text);
        for (int i = 0; i < preparedText.length() - 1; i += 2) {
            if (preparedText.charAt(i) == preparedText.charAt(i + 1)) {
                preparedText.insert(i + 1, 'X');
            }
        }
        if (preparedText.length() % 2 != 0) {
            preparedText.append('X');
        }
        return preparedText;
    }
    private static int[] findPosition(char c) {
        for (int row = 0; row < 5; row++) {
            for (int col = 0; col < 5; col++) {
                if (keyMatrix[row][col] == c) {
                    return new int[]{row, col};
                }
            }
        }
        return null;
    }
}
```

OUTPUT:



```
PS D:\Probin's Work\DSA> javac crns.java
PS D:\Probin's Work\DSA> java crns

Enter the encrypted text: ykixkzskkzotmgzznkvgrgik
Encrypted Text: ykixkzskkzotmgzznkvgrgik
Most Frequent Letter: K
Assumed Shift: 6
Decrypted Text: secretmeetingatthepalace
```

LATEST APPLICATIONS:

- **Secure Communications** – Used in low-resource encryption systems, such as military field encryption and emergency communications.
- **Embedded Systems Security** – Applied in IoT devices for lightweight cryptographic solutions.
- **Data Integrity Checks** – Helps in verifying message integrity in constrained environments.
- **Educational Cryptanalysis** – A useful tool in cybersecurity training for understanding classical cryptographic weaknesses and improvements.

LEARNING OUTCOME:

- Practical Application of Cryptography: Developed encryption and decryption programs in Java, reinforcing concepts of digraph encryption methods
- Understand the working of the Playfair cipher for encryption and decryption.
- Analyze ciphertext to identify patterns and weaknesses in traditional ciphers.
- Implement an extended Playfair cipher to enhance security using a larger character set.

References:

1. https://www.baeldung.com/cs/playfair-cipher
2. https://www.tutorialspoint.com/cryptography/cryptography_playfair_cipher.htm
3. https://www.geeksforgeeks.org/playfair-cipher-with-examples/